Remarks

Status of application

Claims 1-40 were examined and stand rejected in view of prior art. Claims 41-59 were withdrawn pursuant to a Restriction Requirement and are now canceled. The pending claims have been amended to further clarify Applicant's invention. Reexamination and reconsideration are respectfully requested.

The invention

A database system providing self-tuned parallel database recovery is described. In one embodiment, for example, in a database system, a method of the present invention is described for performing recovery operations using an optimal number of recovery threads, the method comprises steps of: (a) spawning an initial recovery thread to perform recovery operations; (b) measuring I/O (input/output) performance with the initial recovery thread; (c) spawning a subsequent recovery thread to perform recovery operations; (d) measuring I/O performance with the subsequent recovery thread; and (e) as long as I/O performance does not degrade beyond a preselected percentage, repeating steps (c) and (d) for spawning a desired number of additional recovery threads.

General

The claims have been amended to address any indefiniteness concerns raised by the Examiner. Accordingly, it is believed that any rejection under Section 112, second paragraph, is overcome.

Prior art rejections

A. Section 103: Lomet, Lahey, and Klotz

Claims 1-40 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Lomet (Patent Number 6,490,594) in view of Lahey et al. ("Lahey" hereinafter) (Patent Number 7,028,303) and further in view of Klotz et al. ("Klotz" hereinafter) (Publication Number 2004/0015762). Here, the Examiner cites a disparate collection of art: Lomet which describes database recovery technique, Lahey which describes spawning threads (under control of a maximum number), and Klotz which describes a test suite that writes

test data to figure out I/O performance. For the reasons stated below, Applicant's claimed invention may be distinguished on a variety of grounds.

Applicant does not claim to have invented the notion of database recovery technique, nor the notion of spawning threads up to a preselected maximum number, nor the notion of writing test data for figuring out/measuring I/O (input/output) performance. Instead, Applicant has invented a database restore technique that is self-tuning. To be sure, Applicant's invention includes features that overlap with some elements from the prior art. However, as shown below, Applicant's claimed invention combines elements in a non-obvious way to provide a self-tuning database restore technique, which tunes itself on-the-fly (i.e., during actual production use) using live or real data (not test data). Applicant's claimed invention also includes certain elements that are not present in the prior art.

Applicant's invention functions to provide dynamic adaptation that is not taught or suggested by the prior art, including the combination of Lomet, Lahey, and Klotz. In particular, Applicant's invention measures I/O performance and uses that result as a determining factor as to whether to spawn additional threads or not. In this manner, Applicant's invention may dynamically adapt to the then-current environment for reaching maximum throughput. Even though Lahey describes a preselected (i.e., static) maximum number of threads to spawn, Lahey lacks Applicant's approach of **dynamically adapting** to the throughput achieved by the current environment, in order to determine what number of threads will be employed.

In order to reproduce this aspect of Applicant's invention, the Examiner needs to modify the particular teaching of Lahey, so that instead of Lahey's static setting (of maximum number of threads) Lahey is somehow adapted to dynamically set maximum threads spawned. However, as the Examiner has already acknowledged, Lahey does not include any mechanism for monitoring throughput (during actual use) that would allow Lahey to dynamically adapt the maximum number of threads spawned. Further, the Examiner suggests that Klotz's I/O measurements could be combined with Lahey to reproduce this aspect of Applicant's invention, however that suggested combination certainly ignores the underlying teachings of each respective reference. Lahey's maximum thread number is preselected (i.e., statically set), thus making it clear that

Lahey's system itself is not set up to receive any sort of input that would allow the maximum to be dynamically set at run time (i.e., during actual system use). If anything, Lahey's teaching of a statically set maximum teaches away from Applicant's dynamic adaptation approach.

Klotz, for its part, provides nothing that would support the Examiner's contention that the Klotz's measurements could be fed back to Lahey's system in a manner that would re-create Applicant's invention. As discussed above, Klotz provides a test suite that writes test data in order to measure I/O performance. The Examiner does not provide any explanation of how Klotz's measurements could be tied to Lahey's maximum thread number. If Klotz's measurements were passed to Lahey's system, one would still have a system (as taught by Lahey) that specifies a maximum thread number based on a preselected or static number (e.g., preconfigured by the user, before it is needed at run time). Further, Applicant's invention performs self tuning using the live or actual data (i.e., the real data from the production database). Measurements of I/O performance are taken as this live data is being processed (i.e., reading data from disk, for restoring the database). This is very different than using a test case or test suite writing test data, such as described by Klotz. If anything, Klotz teaches away from Applicant's approach of measuring I/O performance based on how well the actual production data (i.e., the real data from the database) is being processed (e.g., read or written to disk) by the I/O subsystem as the restore process is actually running.

Importantly, the computing environment in which Applicant's system is deployed (i.e., one or more server computers with network connectivity) is not some sort of static environment. Instead, the actual demands placed on available computing resources in that environment change from moment to moment. For example, while the database is being restored the same server could be servicing other database requests (e.g., from other database engine instances running on the same machine), or could be servicing other Web-related tasks (e.g., servicing connections to a Web server running on the same machine), or could be performing a variety of other computing tasks that are given to server computers. If the server is experiencing heavy demand from Web users, it may be the case that the optimal number of threads for restore is but a single thread (i.e., no additional threads are spawn). In such an environment, therefore, it does not suffice to

simply preselect a maximum number of threads in order to optimize throughput, as the maximum number that is optimal in fact changes from time to time. The time of day (e.g., daytime or nighttime) may itself have a substantial impact on what maximum number of threads yields optimum throughput. Nor does it suffice to measure I/O performance with a one-off run of test data, as the I/O performance will change from time to time depending on I/O resource demands occurring at a given point in time. All told, the resources available in the environment at any given moment are highly variable, and thus optimum throughput is best achieved using an approach that dynamically adapts at run time, when it is actually needed. This level of fine-tuning is not taught or suggested by the cited art.

During development of Applicant's commercial embodiment, the inventors in fact tried as one possible solution statically setting the maximum number of threads that could be spawn, such as a maximum based on using one thread per database engine (e.g., five threads for five database engines). Such an approach (i.e., one that does not use dynamic adaptation) is of course easier to do, as one can avoid all the work required for programming the code logic required to implement dynamic adaptation. However, in practice, the inventors found that the performance with that static approach was worse then simply using a single thread. The reason for this is that a static approach does not take into account resource demands that may occur at run time. Scenarios may occur at run time where spawning one thread per engine will in fact degrade performance, as the system resources may already be tied up for other tasks. In that case, too many threads are spawn and the system's I/O resources simply get swamped. Therefore, a better solution was needed. The inventors provide the solution by creating the claimed self-tuning approach -- that is, system and methodology that self tunes based on the currently running environment. The approach is particularly helpful with database recovery operation, which entails substantial I/O operations for processing database logs.

Applicant's independent claims have been amended to highlight these "dynamic adaptation" and "self-tuning" features of Applicant's invention. For the reasons stated above, it is believed that the claims sets forth a patentable advance over the art. In view of amendments made to the claims (as well as clarifying remarks made above), it is respectfully submitted that any rejection under Section 103 is overcome.

Any dependent claims not explicitly discussed are believed to be allowable by virtue of dependency from Applicant's independent claims, as discussed in detail above.

Conclusion

In view of the foregoing remarks and the amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 408 884 1507.

Respectfully submitted,

Date: January 2, 2007

Digitally signed by John A. Smart
DN: cn=John A. Smart, o, ou, email=JohnSmart@Smart-IPLaw.com, c=US
Date: 2007.01.02 17:20:49 -08'00'

John A. Smart; Reg. No. 34,929
Attorney of Record

408 884 1507
815 572 8299 FAX